1. **Objetivo General**
   Desarrollar una aplicación que permita manejar la descripción del caso expuesto en el punto 4.

2. **Objetivos Específicos**
   2.1. Aplicar los conceptos del modelo conceptual y relacional.
   2.2. Crear una Base de Datos en PostgreSQL.
   2.3. Crear un WebService/REST Service.
   2.4. Crear una página Web.
   2.5. Usar herramientas como PostgreSQL, AngularJs, Bootstrap, HTML5, CSS, y reporting services o cristal reports.
   2.6. Crear un plan de instalación.
   2.7. Crear un plan de proyecto.

3. **Datos Generales**
   3.1. El valor del proyecto: 20%
   3.2. Nombre código: CineTEC
   3.3. La tarea debe ser implementada grupos de 4 personas.
   3.4. La **fecha de entrega:**
      3.4.1. Plan de trabajo: 6/Nov/2018
      3.4.2. Resumen Ejecutivo Avance 1: 8/Nov/2018
      3.4.3. Resumen Ejecutivo Avance 2: 15/Nov/2018
      3.4.4. Resumen Ejecutivo Avance 3: 22/Nov/2018
      3.4.5. Funcionalidad completa: 25/Nov/2018.
   3.5. Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.

4. **Descripción del caso de estudio**
   Con la llegada de la facturación digital por parte del gobierno costarricense la cadena de cines no podrá seguir facturando en papel como lo ha realizado en los últimos 20 años. Por lo tanto la gerencia ha tomado la decisión de desarrollar una nueva aplicación de facturación y venta de boletos.
   La definición de requerimientos se define en la sección 4.1. Es importante mencionar: en caso de retraso de la entrega de la aplicación la empresa contratada deberá pagar la multa impuesta por el gobierno costarricense de $20.000 mensuales por cada semana de atraso en la facturación digital.

4.1. **Requerimientos del Software**

Debe ser desarrollada utilizando AngularJS, Bootstrap, CSS, HTML5, NO se permite utilizar el Entity Framework.

4.1. **Requerimientos funcionales del Sistema**

Aplicación Web Administración:

- Registro de clientes. Donde se almacenará el nombre del cliente, cédula, número de teléfono. Para esta funcionalidad la empresa desea evitar duplicados en su registro único de clientes por lo que solicita utilizar el registro desarrollado por la empresa hermana (TEConstruye).

- Registro de Películas. Donde se almacenará el nombre original de la película, nombre, imagen, duración, protagonistas, director, clasificación.

- Registro de Cines. Deberá poder almacenar un nombre del cine, ubicación, cantidad de salas.

- Registro de Salas. Deberá poder almacenar un identificador, nombre del cine al que pertenece, cantidad de filas, columnas y/o espacios, capacidad.

- Asignación de proyecciones: El sistema debe poder configurar la lista de proyecciones de las películas. Osea debe permitir asignar a una sala, en un momento específico a una película.

Aplicación Web público en general:

- Selección del cine: Inicialmente el usuario debe poder seleccionar el cine elegido.
- Selección de la película: Una vez seleccionado el cine la aplicación deberá mostrar las películas que tiene en cartelera.
- Selección de la proyección: Una vez seleccionada la película el sistema deberá mostrar las proyecciones que tiene para esa película.
- Selección de asiento: Una vez seleccionada la proyección el sistema deberá mostrar todos los asientos (debe diferenciar entre ocupados y desocupados) para la proyección y el usuario seleccionara los asientos deseados.
- Generación de la Factura: Una vez seleccionados los asientos el sistema generara 2 archivos un XML y un PDF con la factura respectiva. El XML se deberá 'enviar' Hacienda (el XML generado debe seguir las especificaciones de hacienda) y el PDF al cliente. Revisar ejemplos anexos.

4.2. **Requerimientos NO funcionales del Sistema**

- El Sistema debe ser una aplicación web (utilizando AngularJs, Bootstrap, HTML5 y CSS).

- La Base de Datos debe estar en postgreSQL.
- La capa de servicios debe estar desarrollada en C# y debe ser desplegada en la nuble AWS/Azure **NO se permite desplegarla en IIS**.
- La Base de Datos debe estar al menos en tercera forma normal.
- Se deben implementar al menos 5 procedimientos almacenados y 2 triggers.

5. **Entregables**
   5.1. Manual de Usuario.
   5.2. Documentación Técnica y del proyecto (descrita en el punto 6).
   5.3. Documento de instalación.
   5.4. Plan de Proyecto.
   5.5. Script de Base de Datos.
   5.6. Script de población de Base de Datos.
   5.7. Aplicación WEB.
   5.8. Web Service/REST Service.
   5.9. Minutas.

6. **Documentación**

   6.1. Se deberá documentar el código fuente.
   6.2. Se deberá entregar un documento que contenga:
      6.2.1. Modelo conceptual utilizando la notación de Chen.
      6.2.2. Modelo relacional.
      6.2.3. Descripción de los métodos implementados.
      6.2.4. Descripción de las estructuras de datos desarrolladas.
      6.2.5. Descripción detallada de los algoritmos desarrollados.
      6.2.6. Problemas conocidos: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
      6.2.7. Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.
      6.2.8. Documentación de evidencia del trabajo en equipo.
         6.2.8.1. Actividades planeadas y su responsable. (Plan de trabajo)
         6.2.8.2. Minutas de sesiones de trabajo. (Seguimiento al plan de trabajo)
         6.2.8.3. Actividades realizadas por cada estudiante. (Bitácora en digital, donde se describen las actividades realizadas, desde reuniones con el compañero de trabajo, investigaciones, consultas, etc. Se debe describir todo por más insignificante que sea, esto demostrará si ustedes están trabajando en realidad. Este es su diario de trabajo, llevan seguimiento de todo en el tiempo, imaginen que si un compañero los releva en su trabajo, le bastaría con leer sus bitácoras para seguir el trabajo.
         6.2.8.4. Evidencia de uso de un manejador de código (se recomienda Gitgub).

6.2.9.Conclusiones y Recomendaciones del proyecto.

6.2.10. Bibliografía consultada en todo el proyecto

6.3. Diagrama de clases y un documento que explique el porqué del diseño.

7. **Evaluación**

1. El proyecto tendrá un valor de un 80% de la nota final, debe estar funcional.
2. La documentación tendrá un valor de un 20% de la nota final, cumplir con los requerimientos especificados en la documentación no significa que se tienen todos los puntos, se evaluará que la documentación sea coherente, acorde al tamaño del proyecto y el trabajo realizado, no escatimen en documentación.
3. Cada grupo recibirá una nota en cada uno de los siguientes apartados Código y Documentación.
4. El profesor no sólo evaluará la funcionalidad del proyecto, esto quiere decir que aunque el proyecto este 100% funcional esto no implica una nota de un 100, ya que se evaluarán aspectos de calidad de código, aplicación del paradigma OOP, uso de herramientas solicitadas, calidad de documentación interna y externa, trabajo en equipo.
5. No se revisarán funcionalidades parciales, ni funcionalidades no integradas.
6. Es responsabilidad de cada miembro del grupo conocer su código, el profesor puede preguntar a cualquier miembro del grupo que le explique alguna funcionalidad/porción de código.
7. De las notas mencionadas en los puntos 1 y 2 se calculará la Nota Final del Proyecto.
8. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
9. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
   9.1. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
   9.2. Si no se utiliza un manejador de código se obtiene una nota de 0.
   9.3. Si no se entrega el punto 4 de la documentación se obtiene una nota de 0.
   9.4. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
   9.5. Si el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
   9.6. Si el grupo no cuenta con los equipos necesarios para realizar la revisión y no avisó al profesor de esta situación obtendrá una nota de 0.
   9.7. El código debe ser desarrollado en C#, en caso contrario se obtendrá una nota de 0.
10. Cada grupo tendrá como máximo 30 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
11. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
12. Cada grupo es responsable de llevar los equipos requeridos para la revisión.
13. Durante la revisión únicamente podrán participar los miembros del grupo, asistentes, otros profesores y el coordinador del área.
14. Las revisiones se realizan con los estudiantes matriculados en el curso, cualquier persona fuera de estos y los mencionados en el punto 14, no pueden participar en la revisión.
15. Después de enviada la nota final del proyecto el estudiante tendrá un máximo de 3 días

hábiles para presentar un reclamo siempre y cuando la funcionalidad esté completa.

8. Referencias

AngularJS (2018-10-04). Recuperado de: https://angularjs.io

Bootstrap Themes & Templates (2018-10-04). Recuperado de: https://wrapbootstrap.com/

How to Write Doc Comments for the Javadoc Tool. (2018-10-04). Recuperado de: http://www.oracle.com/technetwork/articles/java/index-137868.html

C# Coding Conventions (C# Programming Guide). (2018-10-04). Recuperado de: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions

Anexos y estructuras (Ministerio de Hacienda). (2018-10-26). Recuperado de: https://www.hacienda.go.cr/ATV/ComprobanteElectronico/frmAnexosyEstructuras.aspx#